# VxWorks Local Applications

*Scheme for support*

Wed, Feb 3, 1999

Local applications are separately compiled code that can be downloaded to an IRM node. It is dynamically housed in non-volatile memory. Each program must be position independent and have its entry point at the beginning. When a program is called into activation for the first time, a copy is made of the code in dynamically-allocated memory where it will execute. It gets an initialization call to allow it to prepare for subsequent execution. It gets invoked at 15 Hz, or whatever the operating cycle. It may also receive network message calls if it requests them. (It may be a server that waits for datagrams directed to a specific port, for example.) If it is terminated, it gets a terminating call. Each local application is passed a set of parameters, including a pointer to its own context memory that it established during its initialization call. If a new version of a local application is downloaded, and if the local application is already active, the active version is terminated and the new version copied for execution in dynamic memory, then initialized. The effect is that the new version transparently replaces the old one upon downloading.

A local application has an enable control bit, which is its first parameter. There are 9 additional parameters, each specified as a 16-bit word. Often, such parameters are status/control bit#s or analog channel#s. There may be multiple instances of a given local application, say, for a closed loop support. Each instance shares a single common copy of executable code, but each has its own set of parameters and its own allocated context memory. In summary, a local application permits extension of system services. Once downloaded, it can become a permanent addition to the system operation, since the state of its enable bit, the set of parameter values, is preserved across system resets, as well as the code itself. When the system resets, each enabled local application instance is naturally initialized automatically.

The question is, how can this be done under VxWorks?

Page applications are also supported in a similar fashion to local applications. The main distinction of a page application is that it creates its own page interface based upon a simple 16 line by 32 character monochrome display. The display is viewed by a client by simple access to a 512-byte block of text. The cursor row and column are also accessible. Units and mode lights can be turned on/off. The model is that of the original "little consoles." The display screens can be emulated by the client. The page application runs in the target node, and it is interactively controlled by the client using its own user interface.

A page application has 120 bytes of page-private memory that it can access, which serves as a repository for its own context to be preserved across activations. This can be compared with the parameter values that apply to each local application instance. A page application that wants to keep a context during its own execution may allocate memory at initialization time and keep a pointer to this memory in this page-private memory. Multiple pages (range 1–31) may share the same executing code. Only one page application may be active at one time; only a single screen is emulated. Like a local application, a page application may also get network calls. In addition, a page application can get a keyboard interrupt call, which may be thought of as a mouse click. Its significance often relates to the current cursor position.